

Procedural Programming

Lecture 3 - Summer Semester 2014

Prof. Dr.-Ing. Axel Hunger

Faculty of Engineering
Institute of Computer Engineering

HOW DOES COMPUTER
PROGRAMMING WORK?

MAGIC.

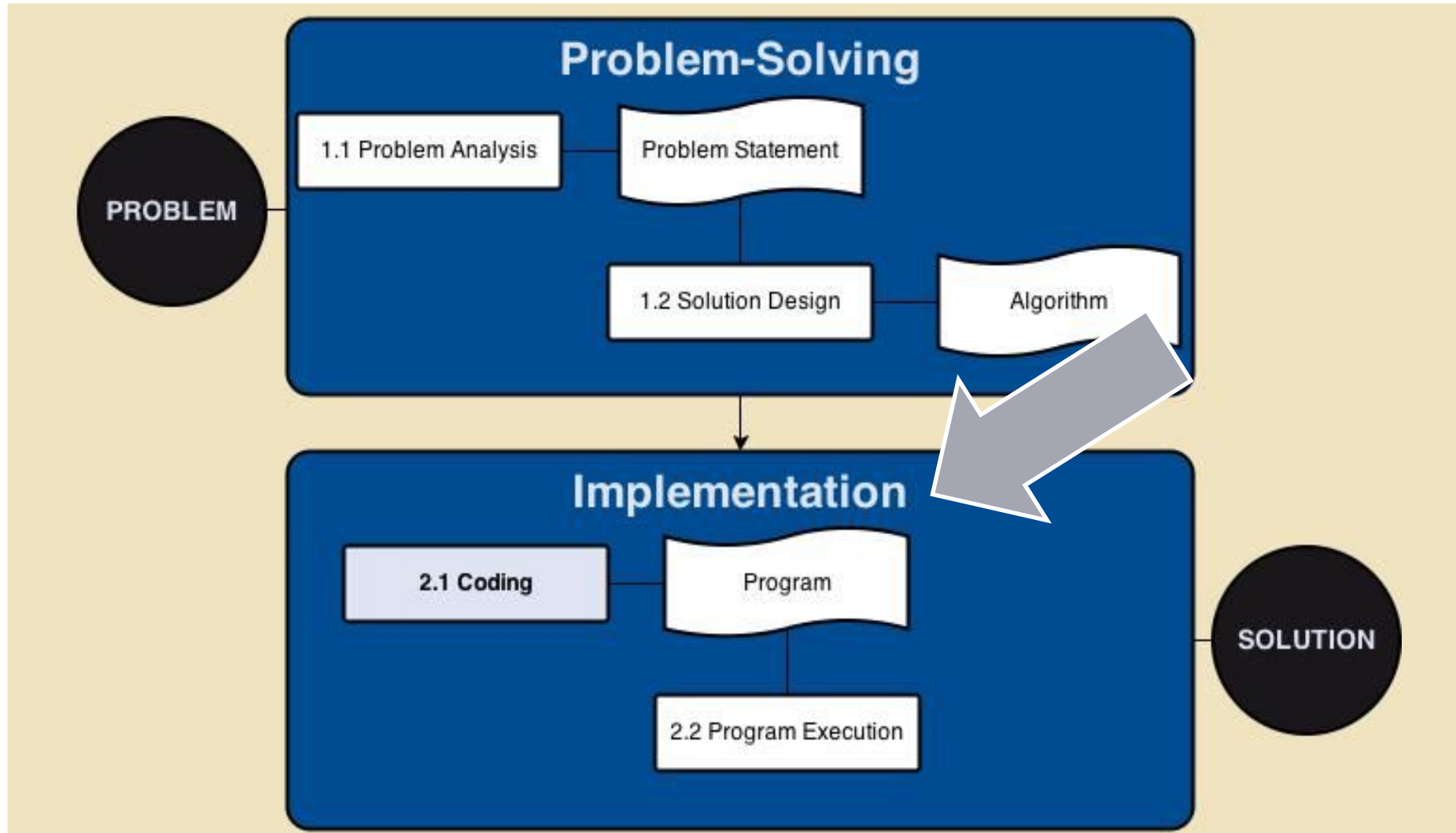


- Effective **problem-solving** requires a comprehensive understanding of the problem situation and a systematic approach in designing a step-wise solution.
- Systematic means we make use of given problem-solving **methods**, such like reduction, and we **document** our solution approach (algorithm) in a commonly accepted way, e.g. flowchart or pseudocode.

“a problem well put is half solved”
— John Dewey

- A computer executes the instructions given in a program one after another. This sequence can be manipulated by use of program **control structures** in order to repeat or skip parts of the program.

Writing a computer program is only one step in the Program Development Process



Pseudocode: How to cross the river?

PROCEDURE RowBoatFromNearToFarShore

REPEAT

LoadItem

UNTIL Safe

CrossTheRiver

UnloadItem

ENDPROCEDURE

PROCEDURE RowBoatFromFarToNearShore

IF Safe

THEN CrossTheRiver

ELSE

LoadItem

CrossTheRiver

UnloadItem

ENDIF

ENDPROCEDURE

START

CALL RowBoatFromNearToFarShore

WHILE NotGoalState **DO**

CALL RowBoatFromFarToNearShore

CALL RowBoatFromNearToFarShore

ENDWHILE

END

START

REPEAT

LoadItem

UNTIL Safe

CrossTheRiver

UnloadItem

WHILE NotGoalState **DO**

IF Safe

THEN CrossTheRiver

ELSE

LoadItem

CrossTheRiver

UnloadItem

ENDIF

REPEAT

LoadItem

UNTIL Safe

CrossTheRiver

UnloadItem

ENDWHILE

END

PROCEDURE

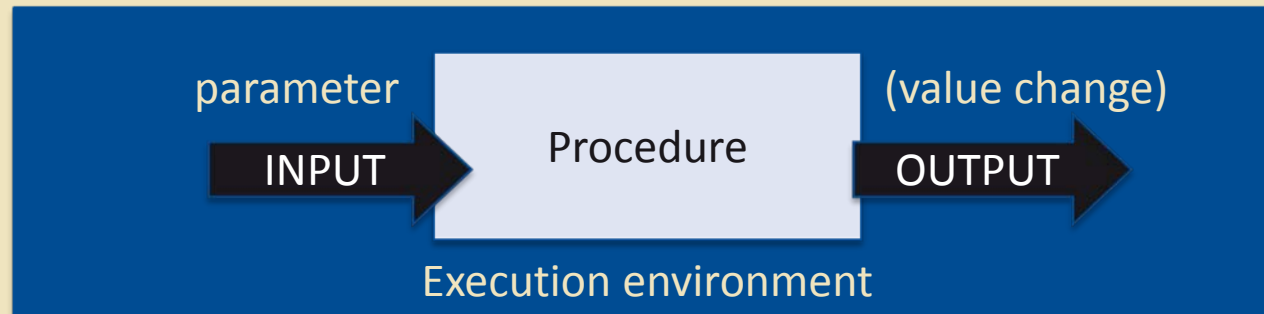
RowBoatFromNearToFarShore

PROCEDURE

RowBoatFromNearToFarShore

Pseudocode without use of procedures.

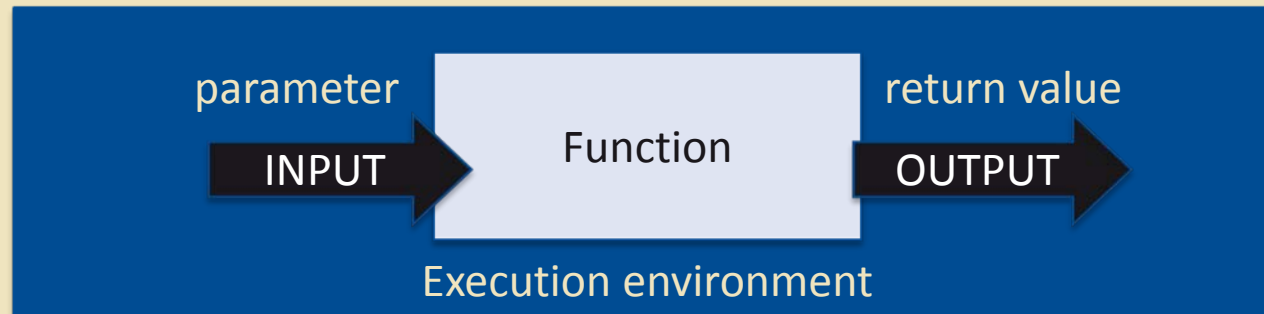
- Procedure contains a sub-sequence of instructions given in the program to be carried out by the computer. It might be called at any point during program execution.
- Procedural programming is based upon the concept of procedure calls.



Procedure

- contains a subset of a program's statements by which it **can** produce change of values.

- Procedures do calculations *without* explicitly returning a value
- Empty return type of procedures is indicated by keyword: void

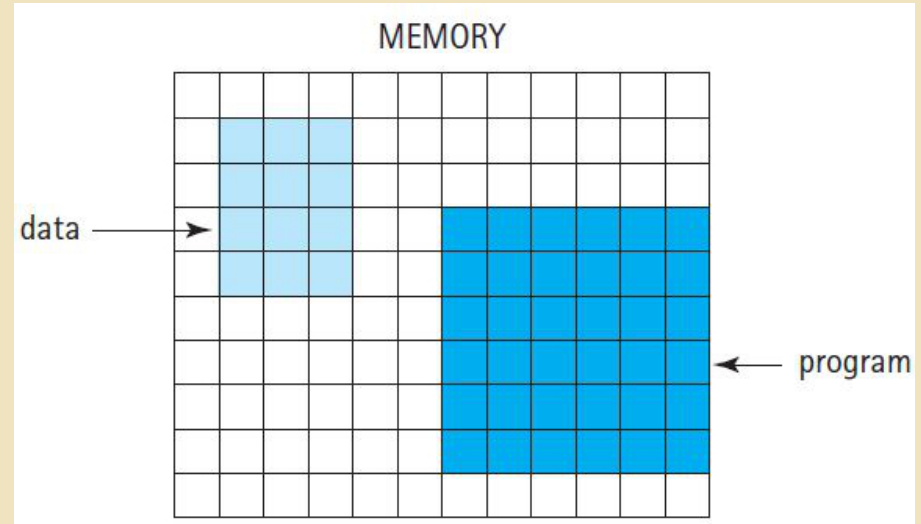


Function

- is a kind of procedure, which calculates a value and returns it as a result of calling this procedure.

- Functions are also known as function procedures
- Return type of a function can be any available data type

- Data is stored in the main memory of the computer.
- The main memory consists of equally sized (and thus in its capacity limited) storage cells, each can be accessed by an individual address.



- In a computer, data is represented by state of electricity (on=1, off=0). Any kind of data can be represented (in a binary form) by a combination of 1s and 0s.
- We have to choose which combination (bit pattern) represents each piece of data.
- For example, 010010101 could represent the word „computer“.

Binary Number with ...	can represent ... different patterns
1 bit	2^1 (2)
8 bits (= 1 Byte)	2^8 (256)
10 bits	2^{10} (1024)

- The smallest addressable unit of data is usually a group of 8 bits called a **byte**.
- The unit processed by machine code instructions is called a **word** (typically 32 or 64 bits).
- Assigning bit pattern to pieces of data is called **coding**, because the only language the first computer recognized was binary in form, where programming meant translating both data and algorithms in patterns of 1s and 0s.
- Today – its the same process we know as programming, where we translate an algorithm into a programming language.

- Patterns of bits by themselves are meaningless. It is the way the patterns are used that gives them meaning.

Data Type

- classifies a set of data (values, which a variable can possess) by defining its characteristics and how to change it.
-
- A data type allows for identifying a particular type of data. It determines
 - possible values for that type,
 - their meaning,
 - the operations that can be done on that values and
 - how these can be stored.

	Integer	Boolean
Meaning or type characteristic	whole numbers	true or false
(range of) Values	-2,147,483,648 to 2,147,483,647	1 or 0
operations	add, sub, mult, mod	set, unset
storage	32 bit	1 bit

- The *internal representation* of data is the way (as a sequence of bits) the value is stored in the main memory of the computer.

○	Adr.	Content of storage cell																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																										
○																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												

Model Attribute

100

A

72

B

...as
represented
in model

Value...

...as externally
represented in
programming
language

```
integer A := 100;
float B := 72.0;
```

- **Variable:** provides temporary storage of data values.
- **Data Type:** defines the type of data that will be stored in a variable.
- By data types a programmer can decide on the storage for a variable that a compiler should reserve.
- A compiler may use the type of a value to optimize the storage it needs and the choice of algorithms for operations on the value.
- Avoidance of type confusion (add of integer is different to add of float).
- Compilation and execution are two distinct processes
 - Compilation translates program to machine language
 - Execution processes machine language version of program

- Every variable must be assigned a data type and given a unique name (identifier).

```
integer A;
```

Declaration of A

- **Declaration of A:** gives information to the compiler about the variable; i.e. its type and its name is **announced**.

```
integer A = 100;
```

Definition of A

- **Definition of A:** **allocates** a memory cell for the variable (e.g. on its initialization).
- It depends on the programming language used to which extent declaration and definition of variables is distinguished.

Declaration and Definition of (function) procedures (1/2)

- A **procedure** introduces a **new statement**. Making use of this statement is named **procedure call**.
- Procedures calling themselves are named recursive procedures.
- **Declaration** of procedure: is the announcement of its signature by naming its **head**: (return) type, name and a list of parameters
<return_type> <name> (<input_param_1>, ..., <input_param_n>)

```
PROCEDURE RowBoatFromNearToFarShore  
REPEAT  
  LoadItem  
UNTIL Safe  
  CrossTheRiver  
  UnloadItem  
ENDPROCEDURE
```

```
void rowBoatFromNearToFarShore(void){  
  ...  
}
```

- **Definition** of (function) procedure is defining its body, which gives the implementation of procedure.

```
void absoluteNumber(int n){
```

```
    if (n>0)
        printf(„Your number is > 0“);
    else
        printf(„Your number is < 0“);
```

```
}
```

head

body

- There are two kinds of parameter:
 - **Formal** parameter, which represent a special kind of variable used in procedure
 - **Actual** parameter, which are a piece of data provided as input to the procedure
- Parameters are the local variables of a procedure, which are initialized by values given on a procedure call.

```
void absoluteNumber(int n);          formal parameter n
int number = -13;
int factor = -1;

main(){
    absoluteNumber(number);          actual parameter number
}

void absoluteNumber(int n){
    int number = n;
    if (number>0)
        then;
    else number = number * factor;
    printf(„Your number is: %d\\n“,number);
}
```

Passing parameters – requires knowing which values to use?

- When calling a procedure, it needs to know which values to use, named actual parameters (or arguments), which can be
 - **explicitly** named values
 - values which are set by **default**
 - **implicitly** given values from the visible environment
- Since the body of a procedure is a self-contained program block, it is possible that there are several variables with the same name in different procedures (or parts of the program) at the same time.
- How does the computer know, which value to use?

- Scoping prevents procedures from assessing the variables of other procedures and vice versa.
- The scope of a variable is the range of statements that can assess it. It is the environment in which the variable is available.
- Within the same scope it is not possible to declare two variables of the same name and type.
- Definitions assign values to an area of memory that was reserved during declaration.

Scope of a variable (2/2)

```
void absoluteNumber(int n);  
int number = -13;  
Int factor = -1;  
  
main(){  
    absoluteNumber(number);  
}  
  
void absoluteNumber(int n){  
    int number = n;  
    if (number>0)  
        then;  
    else number = number * factor;  
    printf(„Your number is: %d\\n“,number);  
}
```

Local environment
with variable number,
whose scope belongs to
the body of procedure
absoluteNumber

Global environment
with variable number

☐ Address	Content of storage cell	
☐		
☐ <u>number</u>	-13	
☐ ...		
☐ <u>number</u>	-13	13

- A variable is visible within its scope or hidden outside it.
- Visibility arises because of the possibility that a variable in outer scope has the same name as a variable in inner scope resulting in 'hiding'.
- Visibility refers to the accessibility of a declared variable.
- Lifetime of a variable is the time in which storage is bound to variable or duration in which the variable is 'alive'.
- Lifetime and visibility of a variable do not have to be the same.

Visibility & Lifetime of a variable (2/2)

variable	scope	visible	lifetime
number	outer	2.-4. and 11.	2.-11.
number	inner	6 to 10	6 to 10

START

```
1. void absoluteNumber(int n);  
2. int number = -13;  
3. Int factor = -1;
```

```
4. main(){  
5.   absoluteNumber(number);  
}
```

11.

END

outer scope

inner scope

```
void absoluteNumber(int n){  
  6. int number = n;  
  7. if (number>0)  
  8.   then;  
  9.   else number = number * factor;  
 10. printf(„Your number is:  
    %d\\n“,number);  
}
```

- Scope is of relevance during compilation (allocation), lifetime is of relevance during program execution (access).
- This holds for variables and procedures.

END
of 3rd Lecture